UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Seminarski rad C

Razvoj igre u Unity razvojnom okruženju

Mentor:
Vladimir Kurbalija

Student:
Natalija Đorđević 81/19

Novi Sad, septembar 2022.

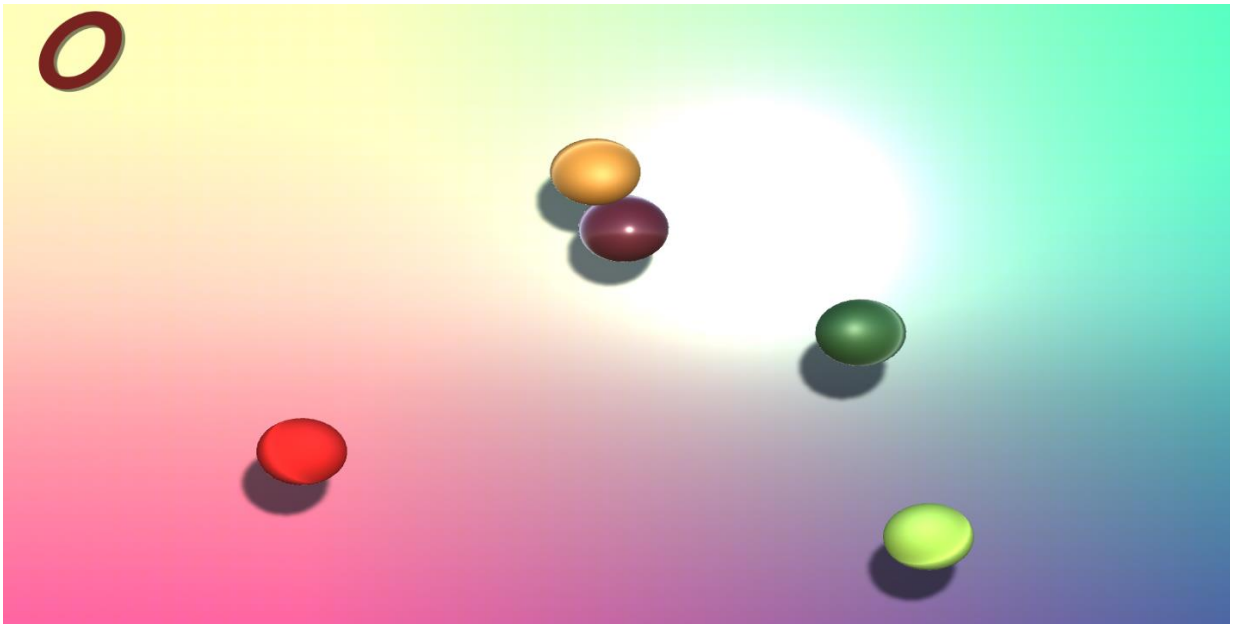
Sadržaj

1. Uvod

U ovom seminarskom radu biće opisano Unity razvojno okruženje, koncepti koji su neophodni za uspešno kreiranje jedne igre, kao i implementacija igre Vegetable killer.

Vegetable killer je jednostavna igra, koja se veoma lako igra i koja može neverovatnom brzinom postati vaša opsesija. Sastoji se od jednog nivoa. Igrač seče povrće sečivom kojim se kontroliše preko ekrana osetljivog na dodir, a može se igrati i na desktop računarima pomoću miša. Dok se voće baca na ekran, igrač prevlači prstom ili klikće mišom preko ekrana da bi napravio pokret sečenja, pokušavajući da iseče povrće na pola. Cilj igre je da se poseče što veći broj različitog povrća kojeg imamo na raspolaganju, gde se pri svakom isečenom povrću naš skor povećava za 1. Ono što trebamo izbegavati jeste bomba, koja se pojavi s vremena na vreme, jer, ako u suprotnom, isečemo i nju, padamo nivo, naš skor se resetuje na nula, i krećemo sa igrom ispočetka.

Na sledećoj slici može se videti kako zapravo izgleda početak igre.



Slika 1. Početak igre

2. Unity razvojno okruženje

Kao što smo malopre pomenuli, ovde će biti reči o Unity razvojnom okruženju.

Unity game engine pokrenut je 2005. godine, sa ciljem da „demokratizuje“ razvoj igara tako što će ga učiniti dostupnim većem broju programera. Sledeće godine, Unity je proglašen za drugoplasiranog u kategoriji najbolje upotrebe Mac OS X grafike u Apple Inc. nagradama za dizajn Apple.

Razvojno okruženje je od tada proširivano kako bi podržalo razne platforme za desktop, mobilne uređaje, konzole i virtuelnu realnost. Posebno je popularno za razvoj igara za Android i iOS uređaje

Unity je prvobitno objavljen za Mac OS X, a kasnije je dodao podršku za Microsoft Windows i web pretraživače.

Unity se može koristiti za kreiranje i razvoj dvodimenzionalnih (2D) i trodimenzionalnih (3D) igara.

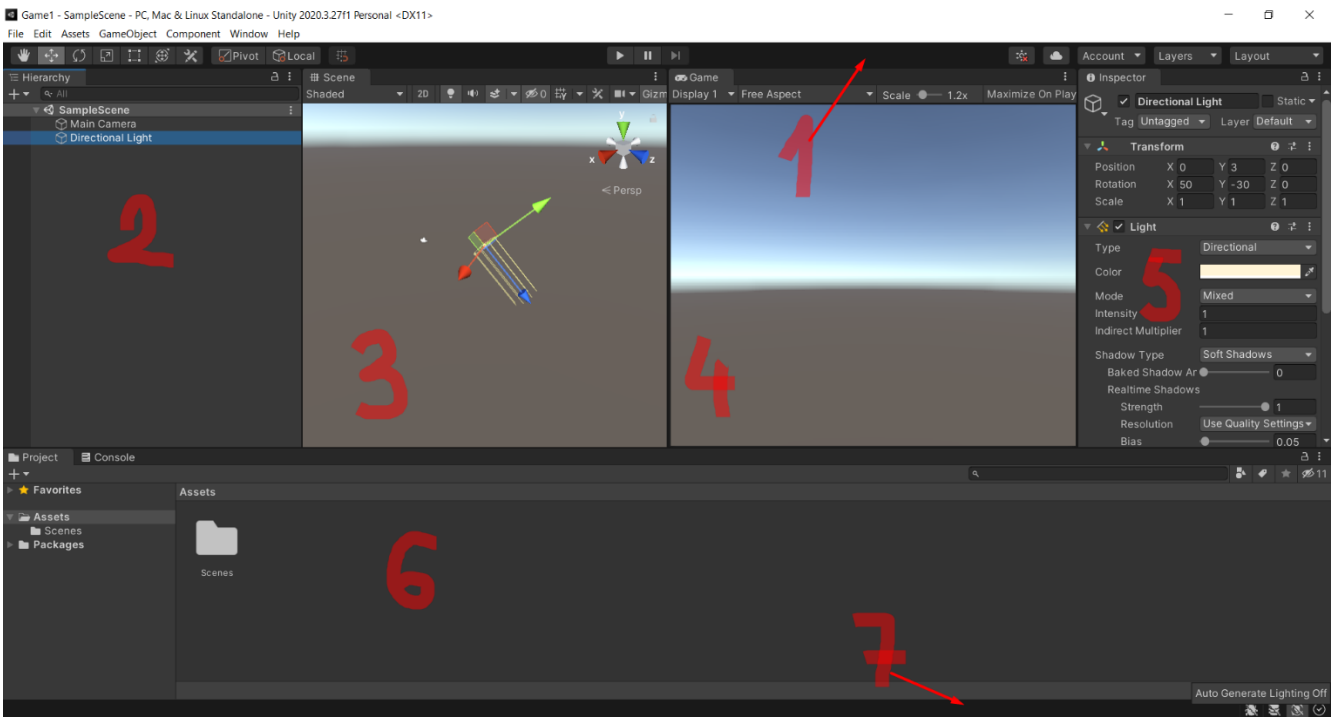
Jedini programski jezik koji je podržan i koristi se za pisanje skripti je C#.

Ima veliku primenu u automobilskoj industriji, a takođe se koristi i u građevinarstvu i arhitekturi.

Danas, imamo 4 različitih verzija Unity – ja koje su izlazile tokom vremena, i koje su nam dostupne. A to su:

- **Unity 2.0** koji je lansiran 2007. sa otprilike 50 novih funkcija. Izdanje je uključivalo optimizovanu mašinu za teren za detaljna 3D okruženja, dinamičke senke u realnom vremenu, usmerena svetla i reflektore, video reprodukciju i druge karakteristike. Izdanje je takođe dodalo funkcije pomoću kojih programeri mogu lakše da sarađuju. Uključuje mrežni sloj za programere da kreiraju igre za više igrača zasnovane na protokolu korisničkih datagrama, nudeći prevođenje mrežnih adresa, sinhronizaciju stanja i pozive udaljenih procedura.

- **Unity 3.0** koji je lansiran u septembru 2010. sa funkcijama koje proširuju grafičke karakteristike motora za desktop računare i konzole za video igre. Pored podrške za Android, Unity 3 je imao integraciju alata Illuminate Labs Beast Lightmap, odloženo prikazivanje, ugrađeni uređivač stabla, izvorno prikazivanje fontova, automatsko UV mapiranje i audio filtere, između ostalog.
- **Unity 4.0** - U novembru 2012. Unity Technologies je isporučio Unity 4.0. Ova verzija je dodala podršku za DirectX 11 i Adobe Flash, nove alate za animaciju pod nazivom Mecanim i pristup Linux pregledu.
- **Unity 5 .0** - Sa Unity 5, motor je poboljšao svoje osvetljenje i zvuk. Preko WebGL-a, Unity programeri su mogli da dodaju svoje igre u kompatibilne web pretraživače bez potrebnih dodataka. za igrače. Unity 5.0 je nudio globalno osvetljenje u realnom vremenu, preglede mapiranja svetla, Unity Cloud, novi audio sistem i Nvidia PhysX 3.3 fizički mehanizam. Peta generacija Uniti engine-a je takođe predstavila efekte kinematografske slike. kako bi Unity igre izgledale manje generično. Uniti 5.6 je dodao nove efekte osvetljenja i čestica, ažurirao ukupne performanse motora i dodao izvornu podršku za Nintendo Switch, Facebook Gameroom, Google Daydream i Vulkan grafički API. Predstavio je 4K video plejer koji je sposoban da pokreće video zapise od 360 stepeni za virtuelnu stvarnost.



Slika 2. Osnovni delovi interfejsa Unity razvojnog okruženja

Nakon instalacije Unity-ja, kreiranja projekta i otvaranja istog, otvara se prozor koji se može videti na slici 2.

Na slici su označeni osnovni delovi interfejsa koji je vidljiv kad se projekat učita u okruženje:

1. Toolbar
2. Hierarchy window
3. Game view
4. Scene view
5. Inspector window
6. Project window
7. Status bar

1. Toolbar predstavlja traku na kojoj se nalaze najbitnije alatke za konfigurisanje same igre, scene i svih objekata u njoj. Sa leve strane se nalaze osnovne alatke za manipulisanje scenom i objektima u njoj. U centru se nalaze tri dugmeta (Play, Pause i Step) koja služe za testiranje trenutno aktivne scene.

2. Hierarchy window je prozor koji služi za prikazivanje hijerarhije svih GameObject-a koji se nalaze na sceni. On se takođe može koristiti i za modifikovanje hijerarhije.

3. Game view predstavlja prozor koji služi za simulaciju naše igre kroz kamere koje se nalaze na sceni. Kada se klikne na dugme Play iz Toolbar-a, simulacija počinje.

4. Scene view je prozor koji prikazuje trenutno aktivnu scenu. Osim prikaza, moguće je vršiti izmene na sceni kao i kretati se.

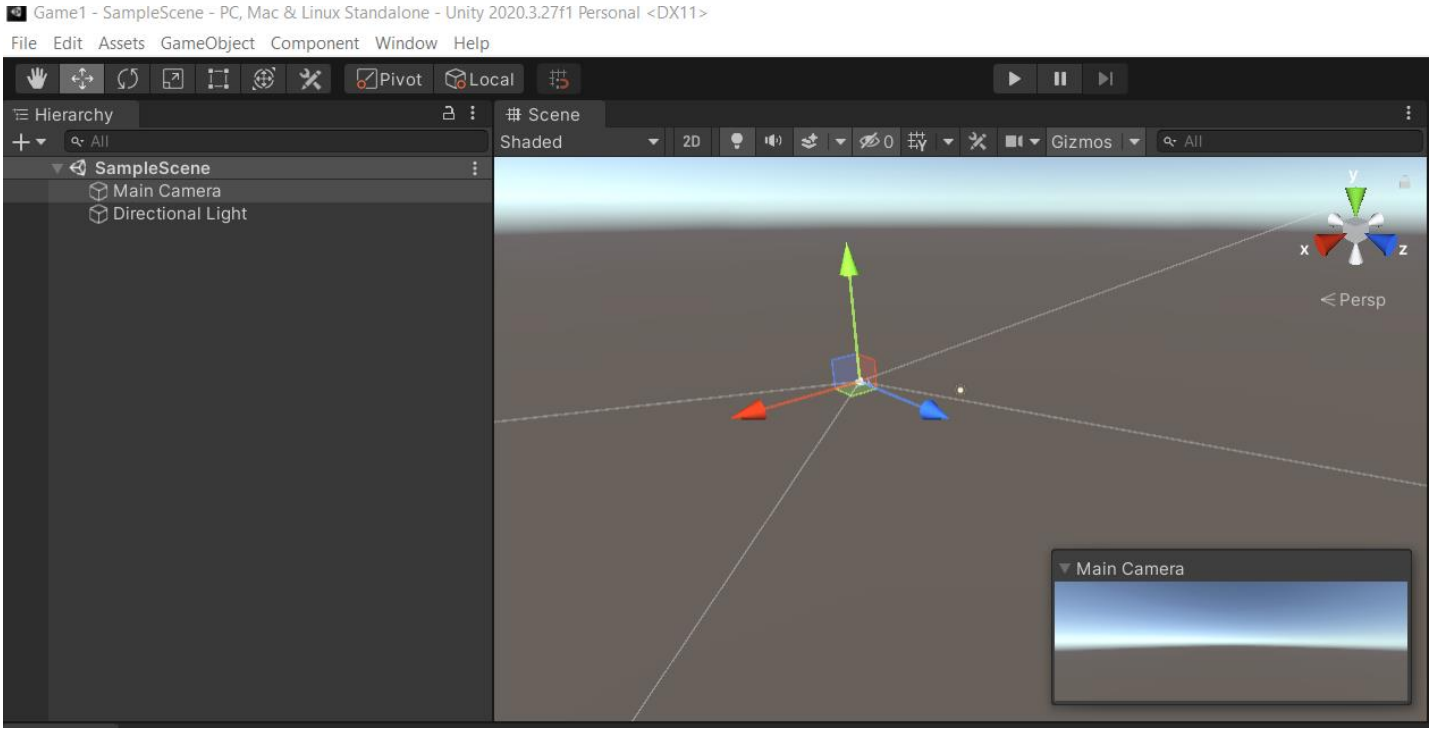
5. Inspector window je prozor unutar kojeg se prikazuju sve komponente i karakteristike izabranog GameObject-a.

6. Project window je prozor u kojem se nalaze svi modeli, skripte, teksture, animacije i sve ono što se napravi ili importuje unutar jednog projekta.

7. Status bar nam pruža obaveštenja o procesima koje Unity trenutno izvršava, kao što su učitavanje projekta, building projekta i slično.

Osnovni koncepti Unity okruženja koji su neophodni za razvoj jedne igre su:

Scene - prostor u koji se smešta sav sadržaj igre. Predstavlja mesto u kome se smeštaju svi karakteri, objekti, kamere, osvetljenja. Kada se napravi Unity projekat, on po default-u sadrži jednu scenu koja sadrži objekat MainCamera koji predstavlja glavnu kameru i DirectionalLight koji predstavlja izvor svetlosti.



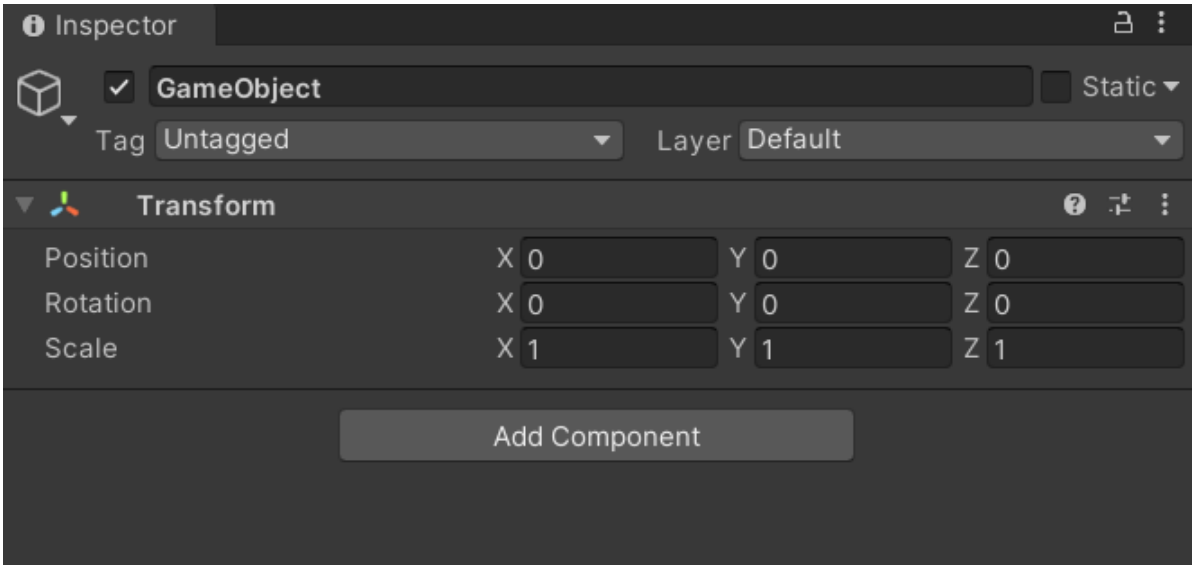
Slika 3. Scena u Unity razvojnom okruženju

GameObject – Predstavlja jedan od najvažnijih koncepata u Unity razvojnom okruženju. Svaki objekat u našoj igri je u osnovi GameObject, od karaktera i svetla do kamera i specijalnih efekata. Oni sami ne postižu mnogo, ali se ponašaju kao kontejneri za komponente koje im daju određenu funkcionalnost.

Constraint - povezuje poziciju, rotaciju ili razmeru GameObject-a na drugi GameObject. Ograničeni GameObject se pomera, rotira ili skalira poput GameObject-a sa kojim je povezan.

Unity sadrži mnogo različitih ugrađenih tipova komponenti za razne potrebe, kao što su komponente za detekciju kolizije, kretanje, fiziku, zvučne efekte, osvetljenje i slično.

Slika 4 nam prikazuje Transform komponentu u okviru GameObject-a.

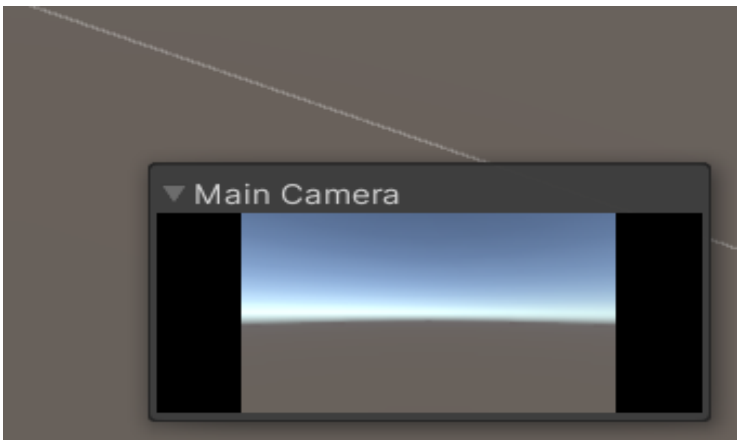


Slika 4. Transform komponenta unutar praznog GameObject-a

Camera - Kamere su jako bitne za igre, jer se ponašaju kao okviri za prikaz, i određuju šta igrač vidi na svom ekranu. Na sceni može biti postavljen veći broj kamera, ali u svakom trenutku je samo jedna glavna - MainCamera koja prikazuje ono što igrač vidi.

Jedan od razloga zašto Unity automatski dodaje MainCamera objekat pri kreiranju novog objekta ili scene je upravo ovaj.

Na slici 5 imamo prikaz Camere u Unity razvojnom okruženju.



Slika 5. MainCamera u Unity okruženju

Canvas - Predstavlja površinu na koju se smeštaju elementi korisničkog interfejsa, kao što su tekst, slike i dugmad. Canvas je GameObject koji sadrži komponentu Canvas, dok su svi elementi korisničkog interfejsa koje želimo prikazati njegova deca. Najčešće se koristi za prikaz statusa igre (na primer za prikaz broja života, trenutni nivo i slično).

Prefab - To je sistem koji omogućava da objekte iz scene, kojima smo podesili Transform i ostale komponente, tako podešene smestimo u projekat, da bismo ih kasnije sa istim tim podešenim komponentama ponovo koristili u našem projektu. Ovo je vrlo korisno za one objekte koji se često pojavljuju na sceni, kao što su prepreke, objekti koji čine samo okruženje, a ponekad čak i karakteri.

Layers - alatka koja nam omogućava da odvojimo i razdvojimo GameObjects u našim scenama. Možemo koristiti slojeve kroz korisnički interfejs i sa skriptama uredimo kako GameObjects unutar naše scene međusobno komuniciraju.

3. Implementacija igre

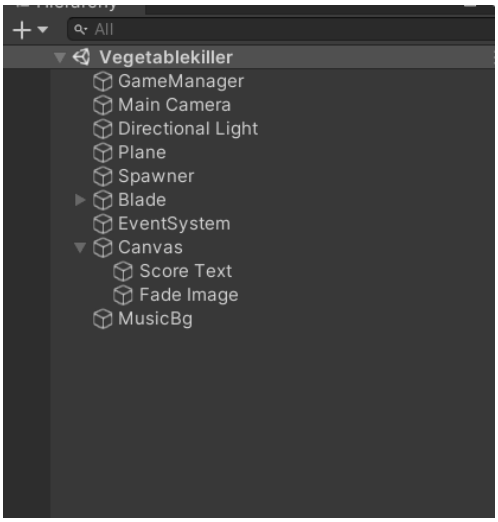
Ovde će biti reči o implementaciji igre Vegetable killer, kao i o istoimenoj sceni. Biće opisana svaka C# skripta koja predstavlja deo igre, ali ćemo prvo pojasniti dva osnovna metoda koja su zajednička za sve skripte, i koja će postojati na skoro svakoj skripti.

- Metod **void Start()** – poziva se u prvom delu pre prvog poziva Update() i tu inicijalizujemo ili dobavljamo iz projekta sve što ćemo koristiti u ostatku skripte.
- Metod **void Update()** - metod koji se izvršava u svakom delu i tu uglavnom stavljamo kod za konstantne provere.

3.1. Scena VegetableKiller

Sada će biti govora o našoj glavnoj sceni VegetableKiller .

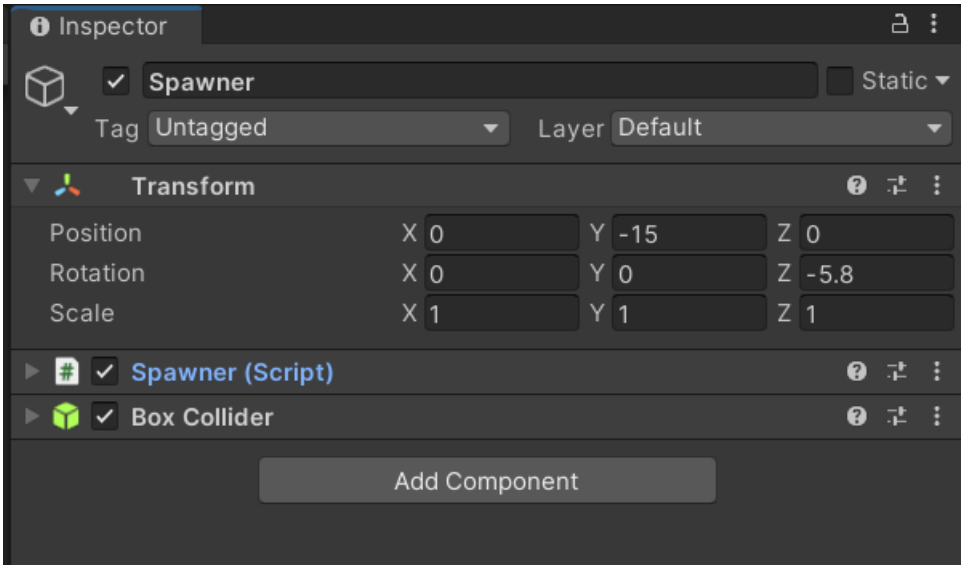
Scena **VegetableKiller** je prva i jedina scena sa kojom se igrač susreće pri pokretanju igrice. U sebi sadrži kameru i osvetljenje koje dobije pri samom kreiranju. Canvas za prikazivanje slika i teksta, kao i EventSystem objekat koji dobije pri kreiranju, GameManager i MusicBg objekte, takođe, na sceni se nalaze i objekti Spawner i Blade što možemo videti na slici br 6.



Slika 6. Hijerarhija scene VegetableKiller

Spawner objekat

Spawner je objekat koji je u osnovi napravljen kao Empty object i koji u sebi sadrži istoimenu skriptu i komponentu Box Collider. U spawner skripti možemo menjati neke vrednosti o kojima će biti reči u sledećem delu, takođe imamo mogućnost i dodavanja povrća kojeg smo kreirali u Prefabs folderu. Slika 7 nam pokazuje strukturu Spawner objekta.



Slika 7. Struktura Spawner Objekta

Skripta Spawner je zaduzena za prikazivanje povrća i s vremena na vreme pojavljivanja bombe na našoj tabli, gde se to povrće seče. U njoj definišemo određene varijable kao što su pokreti povrća, poput rotacije, zatim kako će povrće biti pozicionirano na tabli, koje će biti vreme prikazivanja na ekranu, brzinu kojom će se kretati povrće kao i ugao pod kojim će se pojavljivati.

Skripta sadrži četiri metode: Awake() – metod koji unity automatski pozove kad je skripta inicijalizovana, metos Spawn() koji predstavlja celu logiku igre i ponavljaće se iznova i iznova sve dog se igra ne prekine, metod onEnabled() koji se poziva kada nam je skripta omogućena, i metod onDisabled() koji se poziva kada nam je skripta onemogućena.

```
private void Awake()
{
    spawnArea = GetComponent<Collider>();

}

private void OnEnable(){
    StartCoroutine(Spawn());
}

private void OnDisable(){
    StopAllCoroutines();
}
```

Slika 8. Metodi Awake(), onEnabled(), onDisabled()

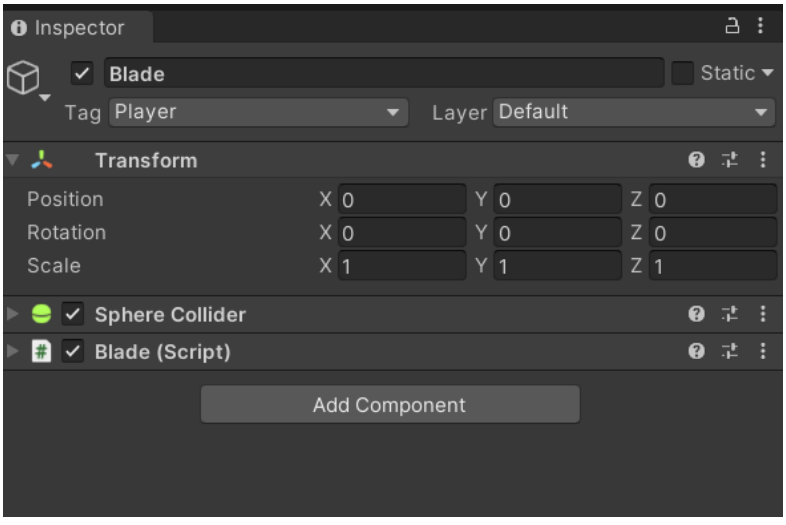
```
private IEnumerator Spawn(){
    yield return new WaitForSeconds(2f);
    while(enabled)
    {

        GameObject prefab= vegetablePrefabs[Random.Range(0, vegetablePrefabs.Length)];

        if(Random.value < bombChance)
        {
         prefab= bombPrefab;
        }

        Vector3 position = new Vector3();
        position.x= Random.Range(spawnArea.bounds.min.x, spawnArea.bounds.max.x);
        position.y= Random.Range(spawnArea.bounds.min.y, spawnArea.bounds.max.y);
        position.z= Random.Range(spawnArea.bounds.min.z, spawnArea.bounds.max.z);

        Quaternion rotation = Quaternion.Euler(0f, 0f, Random.Range(minAngle, maxAngle));

        GameObject vegetable = Instantiate(prefab, position, rotation);
        Destroy(vegetable, maxLifetime);

        float force= Random.Range(minForce, maxForce);
        vegetable.GetComponent<Rigidbody>().AddForce(vegetable.transform.up * force, ForceMode.Impulse);


        yield return new WaitForSeconds(Random.Range(minSpawnDelay, maxSpawnDelay));
    }
}
```

Slika 9. Metod Spawn()

Metod Spawn() kao što smo malopre pomenuli predstavlja logiku igre koja nam na početku omogućava zaustavljanje izvršenja funkcije i mogućnost čekanja da se nešto desi, pre nego što nastavi opet. U našem slučaju 2 sekunde prolazi od kada izgubimo do kada opet krećemo sa novom partijom. Zatim, u while petlji, nasumično biramo povrće koje se prikazuje na ekranu. Pomoću vektora mi zapravo određujemo pozicije našeg povrća, nakon čega slede podešavanja rotacionih uglova. Pomoću funkcije Destroy() uništavamo povrće posle maksimalnog trajanja istog, i na kraju lansiramo povrće na tabli pomoću Rigibody komponete.

Blade objekat

Blade je objekat koji je u osnovi napravljen kao Empty object i koji u sebi sadrži istoimenu skriptu i komponentu Sphere Collider. Blade object ima jedan child objekat – Trail, koji zapravo predstavlja efekat, zapravo on nam definiše kakve će nam oštrica imati pokrete kao i kako će izgledati. Komponenta Sphere Collider nam zapravo određuje kako će oštrica dotaći voće i u osnovi je Sfera. Kod Blade objekta u polju tag selektujemo Player, na osnovu čega znamo da je ono što je dotaklo povrće oštrica a ne neko drugo povrće. Slika 10 nam pokazuje strukturu Blade objekta.



Slika 10. Struktura Blade objekta

Blade skripta zapravo predstavlja oštricu, koja će se koristiti prilikom seckanja povrća. Imamo metod Update(), metod StartSlice(), metod StopSlice() i metod ContinueSlice().

Metod Update() – gde konstantno proveravamo da li se klik miša nalazi gore ili dole gde takođe proveravamo da li smo otprilike u sred držanja miša dole. Ako se klik miša nalazi gore, naše sečivo neće raditi, u suprotnom prikazuje nam se oštrica, a ako je u pitanju treći slučaj tada imamo novi metod za nastavak sečiva. Na slici 11 je prikazan metod Update().

```
private void Update(){

  if(Input.GetMouseButtonDown(0)) {
    StartSlice();
  }else if(Input.GetMouseButtonUp(0)){
    StopSlice();
  }else if(slicing){
      ContinueSlice();
  }
}
```

Slika 11. Metod Update()

Metod StartSlice() , koji pokreće oštricu, gde definišemo poziciju za kameru, koja mora biti u okviru naše table koja je prilagođena isto kao i kod metoda ContinueSlice(). S druge strane imamo metod StopSlice(), koji zaustavlja sečivo, i postavlja ga na false. Na slici 12 je prikazan metod StartSlice i StopSlice().

Metod ContinueSlice() nam zapravo govori da ako nismo upravo počeli i ako nismo stali a usred smo sečenja onda nastavljamo sečenje. Takođe, imamo podešavanje za kameru tako što postavljamo pozicije oštrice na osnovu pozicije miša. Pozicija miša je postavljena tako da on ima mogućnost kretanja po celom ekranu, a nama je cilj da bude u granicama prostora igrice. Ovde takođe određujemo pravac sečenja oštrice. Na slici 13 je prikazan metod ContinueSlice().

```
private void StartSlice(){
  Vector3 position= mainCamera.ScreenToWorldPoint(Input.mousePosition);
  position.z= 0f;

  transform.position= position;

  slicing= true;
  sliceCollider.enabled= true;
  sliceTrail.enabled= true;
  sliceTrail.Clear();

}

private void StopSlice(){
  slicing=false;
  sliceCollider.enabled= false;
  sliceTrail.enabled= false;
}
```

Slika 12. Metode StartSlice() i StopSlice()

```
private void ContinueSlice(){

 Vector3 newPosition = mainCamera.ScreenToWorldPoint(Input.mousePosition);
 newPosition.z= 0f;


 direction= newPosition- transform.position;

 float velocity= direction.magnitude / Time.deltaTime;
 sliceCollider.enabled= velocity > minSliceVelocity;

 transform.position= newPosition;

}
}
```

Slika 13. Metod ContinueSlice()

Skripta Vegetables zapravo sadrži metode za seckanje povrća.

Metoda onTriggerEnter (Collider other) gde proveravamo da li je drugi collider zapravo oštrica ili je u pitanju drugo povrće. Poredimo sa tagom (“Player”) i na taj način znamo da je u pitanju oštrica iz skripte Blade, a ne neko drugo povrće.

```
private void OnTriggerEnter(Collider other)
{
    if(other.CompareTag("Player"))
    {
     Blade blade= other.GetComponent<Blade>();
     Slice(blade.direction, blade.transform.position, blade.sliceForce);
    }

}
```

Slika 14. Metod OnTriggerEnter()

Metoda Slice() nam pokreće seckanje povrća. Kao parametre imamo pravac u koji seckamo povrće, parametar kada nam oštrica dolazi u kontakt sa voćem i snaga koju ima povrće kada ga oštrica preseče.

Definišemo različite stvari poput povećanja scora pri svakom isečenom povrću, rotaciju i uglove seckanja povrća.

```
private void Slice(Vector3 direction, Vector3 position, float force)
{
  FindObjectOfType<GameManager>().IncreaseScore(points);

   whole.SetActive(false);
   sliced.SetActive(true);


   vegetableCollider.enabled= false;
   juiceParticleEffect.Play();

   float angle= Mathf.Atan2(direction.y , direction.x) * Mathf.Rad2Deg;
   sliced.transform.rotation= Quaternion.Euler(0f, 0f, angle);

   Rigidbody[] slices= sliced.GetComponentsInChildren<Rigidbody>();

   foreach (Rigidbody slice in slices)
   {
     slice.velocity= vegetableRigidbody.velocity;
     slice.AddForceAtPosition(direction * force, position, ForceMode.Impulse);

   }
```

Slika 15. Metod Slice()

Game Manager

GameManager objekat je objekat koji sadrzi Text Score i Fade Image koji se odnose na skor naše igre. Sadrži istoimenu skriptu koja objedinjuje sve dosadašnje skripte i pravi celinu igrice.

Metodom Start(), pozivamo metod NewGame(), koji će resetovati skor i krenuti igru ispočetka.

Metod NewGame() – definešemo vreme koje će proći između završene i pokrenute igre,pozivamo metod ClearScene() i blade i spawner omogućavamo.

U metodi ClearScene() prosleđujemo svo naše povrće uključujući i bombu i kada jednom privučemo oštricom na bombu naša partija se završava, čisti se scena i skor se resetuje, a povrće i bomba bivaju uništeni.

Takođe ovde imamo metodu i za povećanje skora. Kao i za eksplodiranjem bombe, koje se desi kada prevučemo oštricom po bombi.

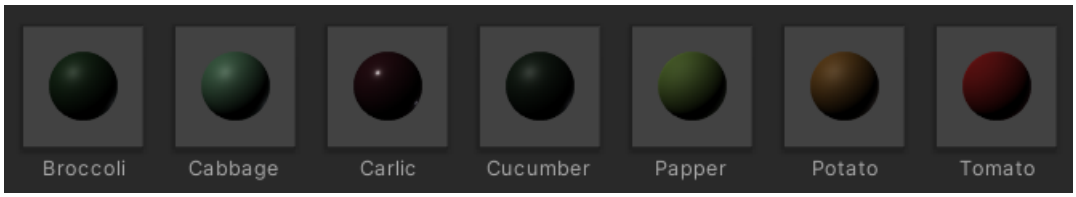
```
private void NewGame()

{
  Time.timeScale = 1f;

  ClearScene();

  blade.enabled= true;
  spawner.enabled= true;

  score=0;
  scoreText.text= score.ToString();


}
private void ClearScene()
{
  Vegetable[] vegetables= FindObjectsOfType<Vegetable>();

  foreach (Vegetable vegetable in vegetables)
  {
      Destroy(vegetable.gameObject);
  }

  Bomb[] bombs= FindObjectsOfType<Bomb>();

  foreach (Bomb bomb in bombs)
  {
      Destroy(bomb.gameObject);
  }
}

public void IncreaseScore(int points)
{

  score+= points;
  scoreText.text=score.ToString();
```

```
public void IncreaseScore(int points)
{

  score+= points;
  scoreText.text=score.ToString();

}

public void Explode()
{
  blade.enabled= false;
  spawner.enabled = false;

  StartCoroutine(ExplodeSequence());

}

private IEnumerator ExplodeSequence()
{
  float elapsed= 0f;
  float duration= 0.5f;
```

Slika 16. GameManager

Vegetable objekat

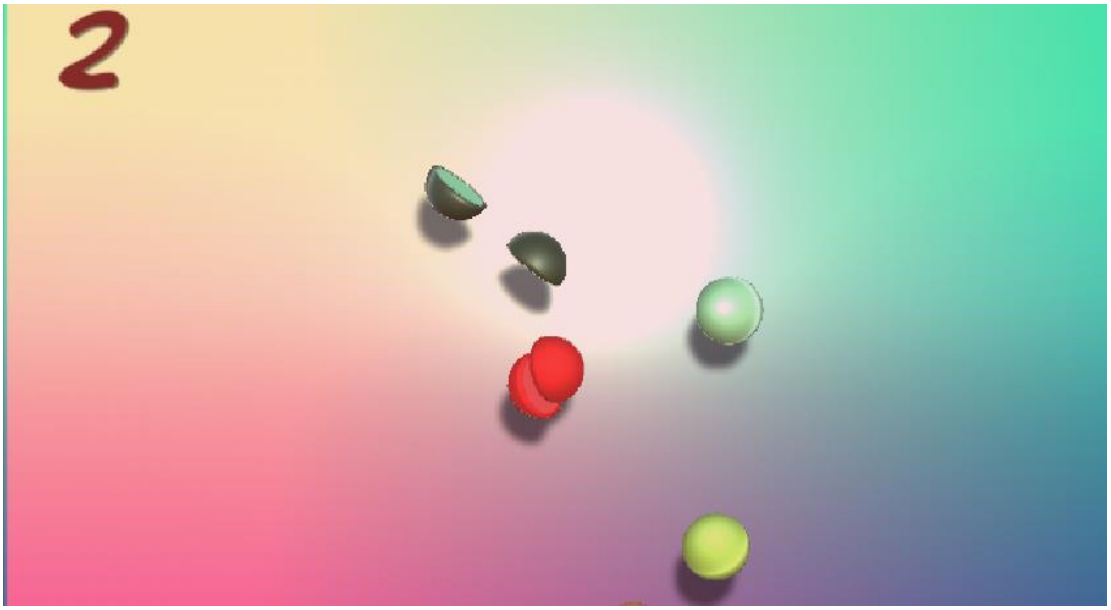
Vegetable objekat je prazan objekat koji predstavlja osnovni material, odnosno bazu za svako povrće koje će se koristiti. Vrste povrća koje u osnovi imaju vegetable objekat možemo videti na slici 17.



Slika 17. Vrste povrća

Canvas objekat

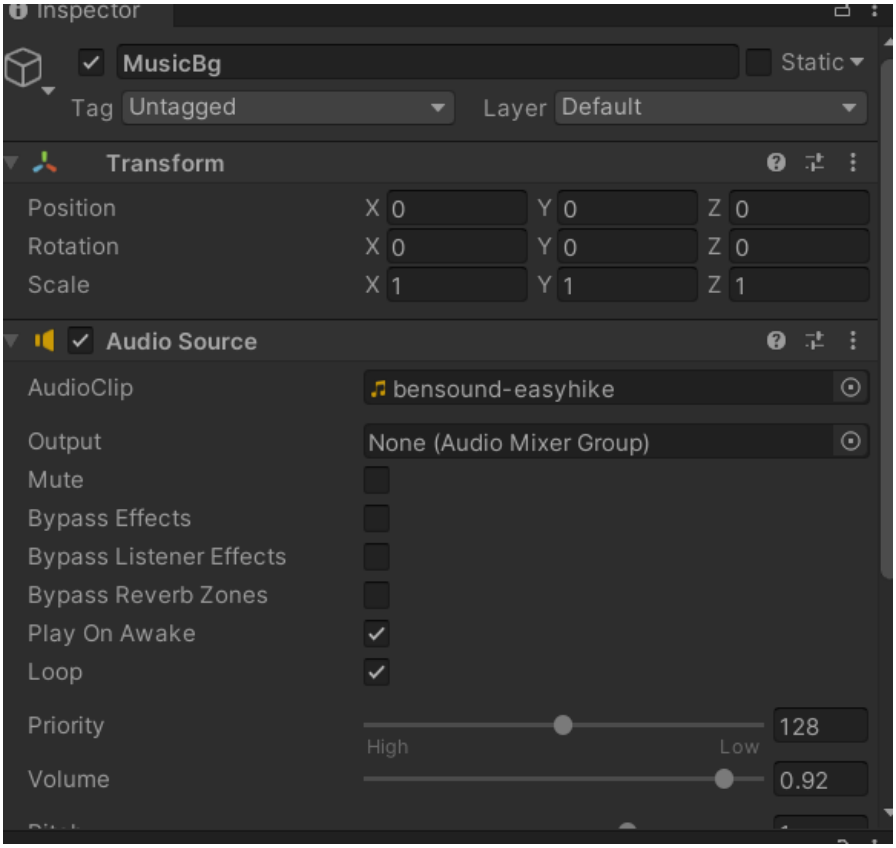
Canvas objekat zadužen je za prikaz Scora, a imamo i sliku koja nam pruža određeni odsjaj kad se igrica pokrene. Na slici 18 se može videti Canvas objekti.



Slika 18. Canvas objekat Score i Image

MusicBg objekat

MusicBg objekat je prazan objekat koji se koristi za upravljanje zvukovima na sceni. Na slici broj 19 možemo videti MusicBg objekat gde smo uvezli AudioClip koji će se puštati pri igranju igrice.



Slika 19. MusicBg objekat

4. Zaključak

Naučila sam razne koncepte koji se koriste prilikom razvoja igara, kao i da koristim samo Unity razvojno okruženje. U početku je bilo jako teško i konfuzno ali se na kraju sve sleglo na svoje. Ova igra nije toliko komplikovana, ali je za mene kao početnika predstavaljala pravi izazov. Nadam se da će mi ovo samo biti odskočna daska u pravljenju još boljih igrica.

5. Literatura

Unity (game engine) - Wikipedia

https://www.youtube.com/c/unity

Unity - Manual: Unity User Manual 2021.3 (LTS) (unity3d.com)